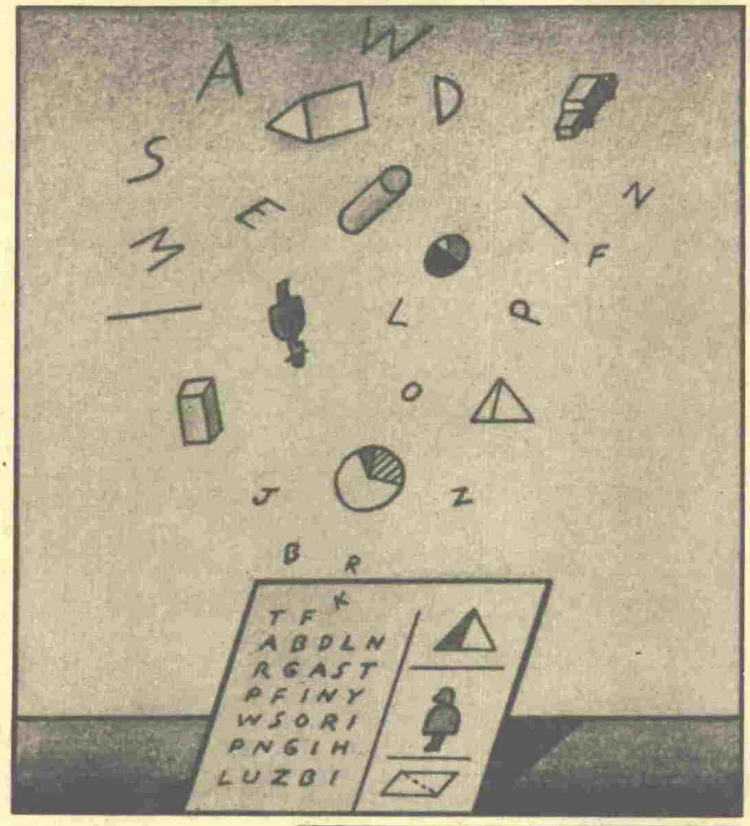
ภาษา LOGØ: โปรแกรมประกอบ



บุญเลิศ เอี่ยมทัศนา

เมื่อผมอ่านคำอธิบายเรื่องของภาษา LOGØ ในหน้าสารบาญในไมโครฯ 48 แล้วรู้สึกอึดอัดใจ เพราะผมไม่มีความสามารถถึงขนาดนั้น ผมเพียงแต่นำเสนอการเขียนอินเตอร์เพรตเตอร์ตามแนวความคิดของผม โดยใช้ภาษาโลโกเป็นตุ๊กตา โดยตั้งความหวังไว้ว่าสักวันหนึ่งเราคงสามารถเขียนโปรแกรมในภาษาโลโกและโปรล็อกได้ด้วยภาษาไทย โครงงานของผมเป็นเพียงตัวอย่างเท่านั้นครับและมีข้อบกพร่องอยู่มากมาย ดังที่ชี้ให้เห็นเป็นระยะ ๆ มา

ในฉบับนี้จะนำเรื่องราวของโปรแกรมที่สร้างขึ้นเป็นเครื่องไม้เครื่องมือสำหรับการเขียนอินเตอร์เพรตเตอร์ต่อไปมาเสนอคือโปรแกรมย่อยในยูนิต CONSOLE ยูนิต CAL กับยูนิต LIBRARY มากล่าวถึง

ที่ผิด

แต่ขอนำจุดที่เป็นข้อบกพร่องของโปรแกรมที่กล่าวไปแล้วในไมโครฯ 48 มากล่าวถึงก่อนคือ

แบบข้อมูล DataT

แบบข้อมูล DataT อยู่ในรูป A2 ตามที่ผมกำหนดไว้ให้มีฟิลด์วนรียนต์เพียงฟิลด์เดียว โดยไม่มีแท็กฟิลด์ เมื่อนำไปใช้มีปัญหาครับเพราะไม่สามารถแยกข้อมูลแบบอักขระกับเลขจำนวนเต็ม ต้องใช้ข้อมูลแบบอักขระในฐานะสตริง และการที่ผมใช้พอยน์เตอร์กับข้อมูลแบบสตริงกับลิสต์ร่วมกันนั้น ปรากฏว่าน่าจะใช้เวลาในการแยกข้อมูลทั้งสองแบบนี้มากกว่าที่จะแยกฟิลด์กัน แต่ในบทความนี้ผมจะยังไม่เปลี่ยนครับ

คำสงวน

ในการกำหนดคำสงวน (รูป A4) ผมได้กำหนดให้ใช้ END; ปิดท้ายโปรแกรมย่อยเต่า ทั้งนี้เพราะผมต้องการตรวจเครื่องหมาย ; เพื่อแสดงว่าจบโปรแกรมย่อยเต่า ดังที่จะเห็นในโพรซีเดอร์ DoConsole และ DoList ในฉบับนี้ ซึ่งความจริงแล้วการทำงานในขั้นตอนนี้ไม่ต้องการความเร็วอย่างยิ่งยวดแต่อย่างใด ทำให้เกะกะเปล่า ๆ ควรใช้แต่คำว่า END หรือต่อไปคือ “จบ” ก็เพียงพอ แต่ผมจะยังไม่แก้อีกเช่นกันครับ

นอกจากนี้ก็มีข้อบกพร่องเล็ก ๆ น้อย ๆ อีก ซึ่งผมจะกล่าวถึงเมื่อถึงโอกาส

ยูนิต CONSOLE

ผมได้รวมคำที่เกี่ยวกับไฟล์และจอภาพที่ผู้ใช้สามารถสั่งได้โดยตรงจากคีย์บอร์ดไว้ในยูนิต CONSOLE ถือ

(เมื่ออ้างถึงรูป A มีหมายเลขต่าง ๆ ขอให้ดูในไมโครฯ 48)

ConsComm

เป็นฟังก์ชันสำหรับตรวจสอบว่าเป็นคำที่ทำหน้าที่คอนโซล ตามที่กำหนดให้แก่ตัวแปร Cons ในรูป A1 หรือไม่ และให้ค่าลำดับที่ของคำกลับไปในตัวแปร def ดังแสดงในรูป C1

DoLoad

เป็นโปรแกรมที่จะโหลดโปรแกรมเต่าลงหน่วยความจำ การทำงานของโปรแกรมนี้เริ่มด้วยการตรวจว่า ผู้ใช้ป้อนชื่อไฟล์ตามหลังคำสั่ง load หรือไม่ ถ้ามีให้นำมาเป็นชื่อไฟล์ ถ้าไม่มีให้ใช้ชื่อไฟล์เป็น TURTLE.LG0

เสร็จแล้วให้เปิดไฟล์อ่านข้อมูลลงหน่วยความจำทั้งหมด ตรวจดูว่าอักขระตัวสุดท้ายเป็น ENDS คือ ; หรือไม่ ถ้าไม่ใช่ให้ปิดท้ายด้วย ENDS กับ ENDF (Ctrl-z)

ในวงรอบ REPEAT เป็นการแยกโปรแกรมย่อยเต่าให้แต่ละ proc[pm] เช่น หากในไฟล์ TURTLE.LG0 มีโปรแกรมย่อยเต่าอยู่ 5 โปรแกรมและใช้ load แต่เริ่มต้น ก็จะทำให้พอยน์เตอร์ของ proc[1] ถึง proc[5] ชี้ไปที่ต้นโปรแกรมเต่าแต่ละโปรแกรมตามลำดับ และเก็บความยาวของแต่ละโปรแกรมไว้ในฟิลด์ len ด้วย ดังแสดงในรูป C2

DoList

โปรแกรมนี้จะนำโปรแกรมย่อยเต่าที่เก็บไว้ในหน่วยความจำตั้งแต่ proc[1] ถึง proc[pm-1] ออกแสดงทางจอภาพทีละโปรแกรม การแสดงในแต่ละโปรแกรมย่อยนั้นจะอ่านอักขระมาแสดงทีละอักขระจนกว่าจะพบเครื่องหมาย ENDS คือ ; จะคอยให้ผู้ใช้กดคีย์ Enter แล้วแสดงโปรแกรมย่อยอื่นต่อไป โดยที่ผมคิดว่าโปรแกรมย่อยเต่าหนึ่ง ๆ คงจะยาวไม่เกิน 1 จอภาพ ดังรูป C3

C1

```
{$V-}
UNIT  Console;

Interface
uses   crt,log00;

FUNCTION   ConsComm(var wrd : String;
                    var def : ConT) : Boolean;
PROCEDURE  DoConsole(def : Cont;
                     px,cx : Word);


Implementation

FUNCTION   ConsComm(var wrd : String;
                    var def : ConT) : Boolean;
VAR    d : ConT;
BEGIN
  ConsComm := True;
  FOR  d := load TO renew DO
    if  wrd = cons[d]  then
      begin  def := d;  Exit  end;
  ConsComm := False;
END;
```

C2

```
PROCEDURE  DoConsole(def : Cont; px,cx : Word);

PROCEDURE  DoLoad(buf : PntrT);
VAR       fi : File;  cfi : FILE OF Char;
          finame : FinameT;
          wrd : IdenT;  len,tem : Word;
BEGIN
  GetToken(px,cx,wrd);
  if  wrd = '' then
    finame := 'TURTLE.LG0'
  else  finame := wrd;
  Assign(cfi,finame);  Assign(fi,finame);
  {$I-}  Reset(cfi);  {$I+}
IF  IOresult <> 0  THEN
  WriteLn('Can''t open file ',finame)
ELSE
  BEGIN
  len := FileSize(cfi);  Close(cfi);
  GetMem(buf,len + 1);
  if  buf = nil  then
    begin  err := 1;  Exit  end;
  Reset(fi,1);
  BlockRead(fi,buf^,len,tem);  Close(fi);
  while  buf^[len] IN [CR,LF,ENDFI]  do  dec(len);
  if  buf^[len] <> ENDS  then
    begin  buf^[len+1] := ENDS;
           buf^[len+2] :=ENDFI  end;
  len := 1;
  while  buf^[len] in [CR,LF]  do  inc(len);
  REPEAT
    tem := len;    proc[pm].pntr := @buf^[len];
    while  buf^[len] <> ENDS  do  inc(len);
    inc(len);
    while  buf^[len] in [CR,LF]  do  inc(len);
    proc[pm].len := len - tem;  inc(pm);
  UNTIL  (buf^[len] = ENDFI);
  proc[pm].pntr := nil;
  END;
END;
```

C3

```
PROCEDURE  DoList;
VAR      p,c : Word;  x,y : Byte;
BEGIN
  temScrn := tranScrn;  OpenWindow(turt);  ClrScr;
  FOR  p := 1 TO  pm-1  DO
    BEGIN
    c := 1;  Write('to ');
    while  proc[p].pntr^[c] <> ENDS  do
      begin  Write(proc[p].pntr^[c]); inc(c)  end;
    WriteLn(proc[p].pntr^[c]);
    x := WhereX;  y := WhereY;
    Write('  press ENTER');
    ReadLn;  gotoXY(x,y);  ClrEOL;  WriteLn;
    END;
  Write('End of database, press ENTER ');  ReadLn;
  OpenWindow(user);
END;
```

C4

```
PROCEDURE  ClearScreen;
BEGIN
  OpenWindow(turt);  ClrScr;  OpenWindow(user);
END;

PROCEDURE  Park;
BEGIN
  col := Xadj;  row := Yadj;
  scrn[row,col].ch := tur;  head := 0;
  rx := 0.0;  ry := 0.0
END;

PROCEDURE  RenewPrg;
BEGIN
  Release(datum);  Init;
  OpenWindow(user);
END;
```

C5

```
BEGIN  { DoConsole }
  CASE  def  OF
    load    : DoLoad(proc[pm].pntr);
    list    : DoList;
    save    : begin  end;
    cls     : ClearScreen;
    prk     : Park;
    renew   : RenewPrg;
  END;  { case }
END;


END.  { unit console }
```

D1

```
{$V-}
UNIT   Cal;
Interface
uses  crt, Log00;

PROCEDURE  DoVariable(wrd : IdenT; px,cx,v1,v2 : Word;
                      var dat : DataT);
FUNCTION  CanGetValue(var px,cx : Word; v1,v2 : Word;
                      var dat : DataT;
                      var ref : Boolean) : Boolean;
PROCEDURE  CalExpress(var px,cx : Word; v1,v2 : Word;
                      var dat : DataT; var res : Word);
```

D2

```
FUNCTION  CanGetValue(var px,cx : Word; v1,v2 : Word;
                      var dat : DataT;
                      var ref : Boolean) : Boolean;
CONST   number : SET OF Char = ['0'..'9'];
VAR   wrd : IdenT;  cc,res : Word;
BEGIN
  CanGetValue := TRUE;  res := 0;  ref := FALSE;
  while  proc[px].pntr^[cx] in skip  do  inc(cx);
  cc := cx;  GetToken(px,cx,wrd);
  IF  fch IN number  THEN  Val(wrd,dat.num,res)
  ELSE IF  fch = LFTPAR  THEN
    begin  cx := cc+1;
           CalExpress(px,cx,v1,v2,dat,res)  end
  ELSE IF  (fch = QUOT) OR (fch = OPNBLK)  THEN
    begin  cx := cc;
           dat.pntr := @proc[px].pntr^[cx];  Exit  end
  ELSE IF  fch = VARSIGN  THEN
    begin  DoVariable(wrd,px,cx,v1,v2,dat);
           ref := TRUE; Exit    end

  ELSE IF  wrd = HEADING  THEN  dat.num := head
  ELSE IF  wrd = ReadCh   THEN
    dat.ch := ReadKey
  ELSE  begin  MarkErr(px,cx,103); res := 300  end;

  IF  res = 0  THEN  Exit
  ELSE IF  res <  300  THEN   MarkErr(px,cx,101);

  CanGetValue := FALSE;
END;
```

ClearScreen

เมื่อผู้ใช้ใช้คำสั่งเต่าไปนาน ๆ บนจอภาพจะเต็มไปด้วยรอยทางเดินของเต่า ผมจึงให้ลบจอภาพได้ด้วยคำสั่ง cls ซึ่งจะมาเรียกใช้โปรแกรมย่อย ClearScreen ดำเนินการลบจอภาพ ดังแสดงในรูป C4

Park

เพื่อนำเต่ามาไว้ตรงกึ่งกลางจอภาพ และให้หันหัวไปในทิศ 000 ดังรูป C4

RenewPrg

เพื่อเริ่มต้นการรันอินเตอร์เพรตเตอร์ใหม่โดยไม่ต้องออกไปที่ดอส โพรซีเดอร์นี้จะยกเลิกโปรแกรมเต่าในหน่วยความจำและเตรียมค่าต่าง ๆ ไว้ที่เริ่มต้นใหม่ทั้งหมด ดังแสดงในรูป C4

โปรแกรมหลักของ DoConsole

ก่อนที่จะเรียกใช้ DoConsole จะต้องตรวจก่อนว่าคำที่จะให้ปฏิบัตินั้นเป็นคำในคอนโซลหรือไม่ ถ้าใช่จะได้ลำดับที่ของคำในตัวแปร def ซึ่งจะต้องผ่านมาให้ DoConsole (ขอให้ดูส่วนหัวของโปรแกรมนี้ในรูป C2) ซึ่งในโปรแกรมหลักก็จะให้ปฏิบัติโปรแกรมย่อยต่าง ๆ ดังกล่าวมา เว้น Save ซึ่งยังไม่ได้เขียนโปรแกรมย่อยให้ ดังแสดงในรูป C5

ยูนิต CAL

ยูนิตนี้ประกอบด้วยโปรแกรมที่เกี่ยวกับตัวแปรและการหาค่าให้แก่ตัวแปร ดังแสดงในรูป D1 ซึ่งมีรายละเอียดดังนี้

CanGetValue

จะหาค่า ณ โปรแกรมย่อยเต่าที่ px อักขระที่ cx ให้แก่ตัวแปร dat ดังแสดงในรูป D2 โดยที่

- ถ้าอักขระตัวแรกเป็นตัวเลขให้เปลี่ยนเป็นเลขจำนวนด้วย Val ซึ่งเป็นโพรซีเดอร์ในเทอร์โบปาสคาล

- ถ้าอักขระตัวแรกเป็น LFTPAR คือเครื่องหมายวงเล็บเปิด ข้อมูลที่ตามมาจะเป็นนิพจน์ เช่น 5+4*3 ให้แปลงค่าเป็นเลขจำนวนด้วย CalExpress

- ถ้าอักขระตัวแรกเป็น QUOT คือเครื่องหมายคำพูดคู่ หรือเป็น OPNBLK คือเครื่องหมายวงเล็บก้ามปูเปิด ข้อมูลที่ตามมาจะเป็นสตริงหรือลิสต์ จึงให้ค่าแอดเดรส ณ ตำแหน่งนี้แก่พอยน์เตอร์ใน dat

- ถ้าอักขระตัวแรกเป็น VARSIGN คือ : ข้อมูลที่ตามมาคือชื่อตัวแปร ให้หาค่าที่เก็บไว้ให้ด้วย DoVariable และให้ ref เป็นจริงเพื่อแสดงว่าเป็นตัวแปร

- หากอักขระตัวแรกไม่ใช่ดังที่กล่าวมา คำ ณ ตำแหน่งนี้ คือชื่อของฟังก์ชันในล็อกศูนย์อันได้แก่ Heading หมายถึงทิศของเต่า และ ReadCh คือให้อ่านคีย์ ให้หาค่าให้ดังที่เห็นในโปรแกรม

หากไม่ใช่ค่าดังกล่าวมาก็หมายถึงมีข้อผิดพลาดเกิดขึ้น ให้หมายข้อผิดพลาดไว้ และให้ res เป็น 300 คือไม่ใช่ 0 เพื่อใช้เป็นเงื่อนไขต่อไป ทั้งนี้เพราะตอนที่แปลงค่าแถวของตัวเลขเป็นเลขจำนวนด้วย Val และการหาค่าของนิพจน์ด้วย CalExpress ผมไม่ได้ตรวจค่า res ว่าดำเนินการได้ถูกต้องหรือไม่ นำมาดำเนินการรวมกันต่อจากนี้

CalExpress

โพรซีเดอร์นี้จะแปลงนิพจน์เป็นเลขจำนวนแต่เป็นโปรแกรมชั่วคราว สามารถดำเนินการได้กับนิพจน์ที่มีเลขเพียง 2 จำนวนเท่านั้น โปรแกรมที่จะใช้งานจริงอยู่ในเรื่อง "โปรแกรมในยุคที่สาม" (ไมโครฯ 36) ซึ่งจะต้องนำมาปรับปรุงอีกเล็กน้อย

การทำงานของโปรแกรมนี้ เริ่มด้วยการหาค่าของเลขจำนวนแรกด้วย CanGetValue มาเก็บไว้ในตัวแปร dat1 อ่านเครื่องหมายในการคำนวณมาไว้ในตัวแปร opr แล้วหาค่าเลขจำนวนหลังให้แก่ dat2 ดำเนินการคำนวณตามเครื่องหมาย ผ่านค่ากลับไปในตัวแปร dat ดังแสดงในรูป D3

DoVariable

โพรซีเดอร์นี้จะหาค่าตัวแปรที่ผ่านชื่อตัวแปรมาทาง wrd ให้แก่ตัวแปร dat โดยตรวจว่าเป็นตัวแปรโลคอลหรือไม่ก่อน ถ้าไม่ใช่จะตรวจว่าเป็นตัวแปรโกลบอลหรือไม่ ถ้าไม่ใช่อีกแสดงว่ามีข้อผิดพลาดให้หมายข้อผิดพลาด

ดังนั้นหากผู้ใช้กำหนดชื่อตัวแปรที่เป็นโลคอลซ้ำกับโกลบอล ก็จะเป็นโลคอลของโปรแกรมย่อยนั้น ๆ เท่านั้น ทำให้สามารถกำหนดชื่อตัวแปรโลคอลซ้ำกับโกลบอลได้ และซ้ำกับโปรแกรมย่อยอื่นได้เพราะค่า U1 และ U2 จะเป็นขอบเขตของโปรแกรมย่อยหนึ่ง ๆ (คือเหมือนกับการกำหนดตัวแปรในภาษาปาสคาล) ดังแสดงในรูป D4

D3

```
PROCEDURE  CalExpress(var px,cx : Word; v1,v2 : Word;
                      var dat : DataT; var res : Word);
VAR    dat1,dat2 : DataT;  ref1,ref2 : Boolean;
       opr : IdenT;
BEGIN
  IF  CanGetValue(px,cx,v1,v2,dat1,ref1)  THEN
    BEGIN
    IF  lch = RGTPAR  THEN
      begin  dat := dat1;  Exit  end
    ELSE
      BEGIN
      GetToken(px,cx,opr);
      IF  (opr[1] IN ['+','-','*','/'])  AND
          CanGetValue(px,cx,v1,v2,dat2,ref2)  THEN
        CASE  opr[1]  OF
          '+' : dat.num := dat1.num + dat2.num;
          '-' : dat.num := dat1.num - dat2.num;
          '*' : dat.num := dat1.num * dat2.num;
          '/' : dat.num := dat1.num / dat2.num;
        END  { CASE }
      END;
    END;
END;

END.
```

D4

```
Implementation

PROCEDURE  DoVariable(wrd : IdenT; px,cx,v1,v2 : Word;
                      var dat : DataT);
VAR    id : Word;
BEGIN
  WHILE  v1 < v2  DO
    BEGIN
    IF  wrd = vars[v1].name  THEN
      begin  dat := vars[v1].data;  Exit  end;
    inc(v1);
    END;

  FOR  id := 1 TO gm-1  DO
    IF  wrd = gvar[id].name  THEN
      begin  dat := gvar[id].data;  Exit  end;

  MarkErr(px,cx,102);
END;
```

ยูนิต LIBRARY

ตามที่เคยกล่าวแล้วว่า ความจริงแล้วยูนิต LIBRARY ก็คือส่วนหนึ่งของยูนิต INTERPRT ที่ผมแยกออกมาก็เพื่อให้เพิ่มคำในภาษาล็อกศูนย์ได้โดยสะดวก

ในยูนิต INTERPRT จะเรียกใช้ฟังก์ชัน StandardIden เพื่อตรวจว่าเป็นคำมาตรฐานหรือไม่ ถ้าใช่จะเรียกใช้ DoStandard เพื่อดำเนินการให้เป็นไปตามคำสั่งของผู้ใช้ โปรแกรมย่อย

หลักของยูนิตนี้จึงมีเพียงเท่านี้ ดังที่แสดงส่วนหัวไว้ในรูป E1 แต่มีโปรแกรมซึ่งเป็นคำในไลบรารีที่จะต้องสร้างขึ้นอีกมากมาย ที่ดำเนินการไว้แล้วเป็นเพียงส่วนน้อยเท่านั้นคือ

CalXY คำนวณหาจุดโคออร์-ดิเนต x,y ที่สัมพันธ์กับตำบลที่เดิมด้วยค่ามุมและระยะทาง (เป็นก้าว) ดังแสดงในรูป E2

Move เลื่อนเต่าไปในทิศหัวของเต่าขณะนั้นเป็นระยะทาง Num ซึ่งจะตรวจด้วยว่าให้เลื่อนไปข้างหน้าหรือถอยหลัง และจะตรวจว่า fast เป็นจริงคือให้กระโดดหรือไม่ ถ้ากระโดดจะหาตำบลที่สัมพันธ์ตามระยะทางที่ให้แก่ num ถ้าไม่กระโดดจะหาระยะทางทีละ 1 (ก้าว) ในการเลื่อนเต่าจะหมายรอยเท้าไว้ ณ ตำแหน่งเดิม แล้วเลื่อนเต่าไป ถ้าตกหน้าต่างจะให้ติดอยู่ตรงนั้น ดังแสดงในรูป E3

Turn ให้หันหัวเต่าไปทางขวาหรือซ้ายตามค่าที่ผ่านให้แก่ idn ด้วยมุมตามที่ผ่านให้แก่ num ดังรูป E4

DoPrint พิมพ์ข้อมูลออกทางจอภาพ โดยตรวจว่า

- ถ้าอักขระแรกเป็นเครื่องหมายคำพูด แสดงว่าเป็นสตริง (ในภาษาล็อกศูนย์ประกอบด้วยแถวของอักขระที่ไม่เว้นวรรค) จึงให้ค่า endstrm เป็นวรรค
- ถ้าอักขระแรกเป็นวงเล็บก้ามปูเปิด แสดงว่าเป็นลิสต์จึงให้ endstrm เป็นวงเล็บก้ามปูปิด
- นอกจากนี้แสดงว่าเป็นเลขจำนวน ให้ตรวจว่าเป็นเลขจำนวนเต็มหรือจำนวนจริง แล้วให้เอาต์พุตโดยตรง

ท่อนต่อไปเป็นการนำอักขระออกแสดงทีละอักขระ เสร็จแล้วให้ตรวจดูด้วยว่า ข้อมูลที่ให้นำออกแสดงเป็นตัวแปรหรือค่าคงตัว ถ้าเป็นค่าคงตัวให้เลื่อน cx ไปไว้ท้ายข้อมูล ดังรูป E5

ClearScreen เพื่อให้ลบจอภาพในโปรแกรมเต่าได้ และเลื่อนเต่ามาไว้

E1

```
{$V-}
UNIT  Library;

Interface
uses  crt,Log00,console,cal;

FUNCTION  StandardIden(var wrd : String;
                       var idn : ComT) : Boolean;
PROCEDURE  DoStandard(idn : ComT;var px,cx : Word;
                                    v1,v2 : Word);
```

E2

```
Implementation

PROCEDURE  CalXY(ang,dst : Real; var x,y : Real);
VAR      rad : Real;
BEGIN
   rad := (90 - ang) * mypi;
   x := dst * Cos(rad);    y := dst * Sin(rad);
END;
```

E3

```
PROCEDURE  Move(idn : ComT; num : Real);
CONST  scale = 1;
VAR    h,x,y : Real;  i,n  : Integer;
BEGIN
  h := head;
  if  idn = bwrd  then
    begin  h := h + 180;  if  h > 360
           then  h := h - 360  end;
  IF  fast  THEN
    begin  CalXY(h,num,x,y);  num := 1  end
  ELSE  CalXY(h,1,x,y);
  FOR  i := 1 to Trunc(num)  DO
    begin  scrn[row,col].ch := step;
           rx := rx + x;
           col := Trunc(rx * scale) + Xadj;
           ry := ry + y;
           row := Yadj - Trunc(ry);
           if col > 80 then  col := 80;
           if col < 1  then col := 1;
           if row > 21 then  row := 21;
           if row < 1  then row := 1;
           scrn[row,col].ch := tur;
    end;
END;
```

E4

```
PROCEDURE  Turn(idn : ComT; num : Real);
BEGIN
  if  idn = lft  then  head := head - num
  else  head := head + num;
  if  head >= 360  then  head := head - 360
  else if head < 0  then  head := 360 + head;
END;
```

E5

```
PROCEDURE  DoPrint(dat : DataT;
                   var cx : Word; ref : Boolean);
VAR   id : Word;  endstrm : Char;
BEGIN
  OpenWindow(turt);  id := 2;
  if  dat.pntr^[1] = QUOT  then
     endstrm := SPACE
  else if  dat.pntr^[1] = OPNBLK  then
     endstrm := CLSBLK
  else  begin
        if  Frac(dat.num) = 0  then
          Write(Trunc(dat.num))
        else  Write(dat.num:3:2);
        OpenWindow(user);  Exit;
        end;

  while  (dat.pntr^[id] <> endstrm) and
         (dat.pntr^[id] <> CR)  do
    begin  Write(dat.pntr^[id]);  inc(id)  end;
  if not ref then  cx := cx + id;
  OpenWindow(user);
END;
```

กลางจอ ดังแสดงในรูป E6

สำหรับโปรแกรมย่อยที่เป็นหลักของยูนิตนี้คือ

StandardIden

เป็นฟังก์ชันที่ตรวจดูว่าคำที่ส่งมาให้แก่ wrd เป็นคำมาตรฐานหรือไม่ ถ้าใช่จะผ่านลำดับที่ของคำนั้นกลับไปในตัวแปร idn ดังแสดงในรูป E7

E6
```
PROCEDURE  ClearScreen;
BEGIN
  OpenWindow(turt);  ClrScr;
  col := Xadj;  row := Yadj;
  scrn[row,col].ch := tur;
  head := 0.0;  rx := 0.0;
  ry := 0.0;
  OpenWindow(user);
END;
```

DoStandard

เมื่อผมเขียนโปรแกรมในยูนิตนี้ในชั้นแรก ผมได้เขียนโพรซีเดอร์ DoStandard เป็นดังรูป E8 แต่ต่อมาผมต้องการลดการเรียกตนเอง ในวงรอบ REPEAT เพื่อประหยัดเนื้อที่ในสแต็กและเวลา จึงเปลี่ยนชื่อเป็นโพรซีเดอร์ Standard แล้วเขียน DoStandard ขึ้นใหม่ครอบ Standard ไว้อีกชั้นหนึ่ง

ก่อนที่จะเรียกใช้โปรแกรมนี้ผู้เรียกจะต้องตรวจก่อนว่าเป็นคำมาตรฐานหรือไม่ (ด้วยเหตุผลที่จะกล่าวในยูนิต INTERPRT) ดังนั้น DoStandard จึงให้ทำคำสั่งนั้นด้วย Standard ไปครั้งหนึ่งก่อน แล้วตรวจว่าคำต่อไปเป็นคำมาตรฐานอีกหรือไม่ ถ้าใช่จะดำเนินการซ้ำในวงรอบ REPEAT จนกว่าพบคำที่ไม่ใช่คำมาตรฐาน หรือเครื่องหมาย] ซึ่งเป็นเครื่องหมายปิดท้ายวงรอบ repeat ของภาษาล็อกศูนย์

การดำเนินการเช่นนี้ทำให้งดการเรียกตนเองได้ทั้งหมด หากในวงรอบ repeat ของภาษาล็อกศูนย์มีแต่คำมาตรฐาน หรืองดได้บางส่วนหากมีคำใหม่ผสมอยู่ด้วย และงดการเรียกตนเอง หากเขียนคำมาตรฐานติดกันถึงแม้จะไม่อยู่ในวงรอบ repeat

ทั้งนี้เป็นไปตามหลักการว่า การเรียกตนเองจะใช้เมื่อจำเป็นเท่านั้น และพยายามหลีกเลี่ยงการเรียกตนเองเมื่อมีโอกาส ที่ทำได้เฉพาะคำมาตรฐานเพราะว่าเป็นคำสำเร็จรูปแล้ว ไม่มีถ้อยแถลงในภาษาล็อกศูนย์ให้แปลอีก จึงไม่ต้องการการเรียกตนเอง

E7
```
          (* ..... Standard Identifier ......... *)
FUNCTION  StandardIden(var wrd : String; var idn : ComT) : Boolean;
BEGIN
  idn := penup;
  while  (wrd <> com[idn]) and (idn <> none)  do  idn := succ(idn);
  StandardIden := (idn <> none);
END;
```

E8
```
PROCEDURE  DoStandard(idn : ComT;var px,cx : Word;
                                    v1,v2 : Word);
VAR    wrd : IdenT;  done : Boolean;  cc : Word;
BEGIN
  done := FALSE;
  REPEAT
    Standard(idn,px,cx,v1,v2);  cc := cx;
    IF  lch <> CLSBLK  THEN
      BEGIN  GetToken(px,cx,wrd);
             if  not StandardIden(wrd,idn)  then
               begin  done := TRUE; cx := cc  end;
      END;
  UNTIL   (lch = CLSBLK) OR done;
END;

END.
```

E9
```
PROCEDURE  Standard(idn : ComT;var px,cx : Word;
                                    v1,v2 : Word);
VAR   dat1,dat2,dat3 : DataT;  wrd : IdenT;
      ref : Boolean;
BEGIN
  IF  idn > nop  THEN
    BEGIN
    IF  not CanGetValue(px,cx,v1,v2,dat1,ref)  THEN
      Exit;
    END;

    CASE  idn  OF
      fwrd,bwrd : Move(idn,dat1.num);
      lft,rght  : Turn(idn,dat1.num);
      print     : DoPrint(dat1,cx,ref);
      penup     : step := nofoot;
      pendn     : step := foot;
      draw      : fast := FALSE;
      nodraw    : fast := TRUE;
      nl        : begin  OpenWindow(turt);
                         WriteLn;
                         OpenWindow(user)  end;
      clean     : ClearScreen;
      nop       : begin  end;
    END;  { case }
END;
```

Standard

เป็นโพรซีเดอร์ที่จะดำเนินการตามคำมาตรฐาน ซึ่งผู้เรียกจะผ่านลำดับที่ของคำมาทาง idn นำมาตรวจดูว่ามีค่ามากกว่า nop หรือไม่ เพราะในการเรียงลำดับคำได้ให้คำที่ไม่ต้องการค่าพารามิเตอร์อยู่ก่อน nop ส่วนคำที่อยู่หลัง nop ต้องการค่าพารามิเตอร์ 1 ตัว จึงให้หาค่าซึ่งถ้าหาไม่ได้แสดงว่ามีข้อผิดพลาดให้ออกจากโปรแกรมนี้ ดังรูป E9

การทำงานในขั้นตอนต่อไปคือนำลำดับที่มาตรวจแยกด้วย CASE ให้ทำงานตามคำนั้น ๆ ด้วยโปรแกรมที่เป็นคำในไลบรารีดังกล่าวแล้ว

โปรแกรมที่สร้างขึ้นเป็นเครื่องมือในการเขียนอินเตอร์เพรตเตอร์ เว้นเอดิเตอร์มีเพียงเท่านี้ ในตอนต่อไปผมจะนำการเขียนอินเตอร์เพรตเตอร์มากล่าวถึง ส่วนเอดิเตอร์ตั้งใจจะเขียนในกราฟิกโหมดโดยตรง ดังจะได้นำพื้นฐานการไปสู่เอดิเตอร์มาเสนอในเรื่อง "ภาษาไทยในกราฟิกโหมด"